CS 461
Fall 2021
Program 1 – Monte Carlo simulation

For our first program of the semester we're going to use a simple simulator to estimate hands in a card game.

The game of Bridge is a partnership game for 4 players. Partners bid for the right to name the trump suit of that hand (or specify that there is none), with the goal of winning the game. Bidding for bridge is fairly complicated—there are several systems out there, well-documented and described—but we're going to use a different approach to estimate the quality of a hand—simulation.

Each player is dealt 13 cards. Thus, each player knows the location of the 13 cards in their hand, and the location of the other 39 cards is a mystery. Each player can estimate the strength of their own hand based on *high-card points* and *distribution* points. These are counted as follows:[1]

High cards:
- Each Ace: 4 points
- Each King: 3 points
- Each Queen: 2 points
- Each Jack: 1 point

Distribution:
- Each doubleton (exactly 2 cards in a suit): 1 point
- Each singleton (exactly 1 card in a suit): 2 points
- Each void (no cards in a suit): 5 points

Your program will shuffle a deck of cards, deal 13 cards for the player's hand, and then repeatedly deal their partner 13 of the 39 remaining cards as one possible hand. You will then estimate the likely best outcome of that hand using the following approximate scale, based on the total points of the 2 hands.
Pass: Less than 20 points
Part score: 20-25 points
Game: 26-31 points
Small slam (take all tricks but 1): 32-35 points
Grand slam (take all tricks): 36+ points.

Your program will deal a large number of simulated hands for the partner (at least 500) and tally the number of times each of these outcomes occurs; it will then print the hand and report this as a percentage falling into each category, as an estimate of the likely outcomes. (That is, if we are running 1000 simulated hands and our point total falls into 26-31 for 521 of them, we report the probability of having a Game as 52.1%).

Note that Bridge is an example of an *imperfect information* game, where some information about the hands is hidden from each player. Much of bidding strategy is about uncovering or providing information about the distribution; of course, the opponents get the same information the partner does.[2]

---

1 See the appendix for a summary of the standard deck if you're not familiar with it.

2 For example, an opening bid of 1 in a major suit (spades or hearts) implies at least 13 high-card points and at least 5 cards of the suit bid; an opening bid of 1 No-Trump indicates 15-17 points and a balanced distribution; etc. If we were writing a bridge-playing program, we'd need to code up those rules; but we're not, so we don't.

We're not going to worry about the complexities of bidding, just get a rough estimate of the strength of this hand.

Your program should deal a hand, print it, run the simulation, and report results. Ask the user whether to continue. The program should continue to run as many times as the user desires.

Programming notes:

- You may write your program in C, C++, C#, Java, or Python.
- For this program, the random number generator (random() function, etc) that comes with the language will be fine. Most modern languages and development environments use the Mersenne Twister or a cryptographic method to generate random numbers, or call a hardware generator. The internet is full of pages warning about the poor quality of the Visual Studio random number generator, which are mostly out of date. It used to be bad[3]; it's been addressed.
  - Note that for some security applications the Mersenne Twister or a cryptographic method *isn't* enough, unless there's a source of entropy being injected regularly, but we're not generating session keys here; we're shuffling some cards. Unless the random number generator for your language is truly a botch, there's nothing we need to worry about here.
- Submit your source code, zipped project folder, or GitHub link to Canvas by the posted deadline.

```
Microsoft Visual Studio Debug Console
Here is your hand:
KS QS 9S 2S AH KH 7H 6H 5H 4H 9D 5C 3C
This hand is worth 15 points.
Running simulation.....

The estimated probability based on 500 simulated hands:
Pass: 5.4%
Part score: 39%
Game: 45.4%
Small Slam: 7.8%
Grand Slam: 2.4%

Another hand [Y/N]? y
Here is your hand:
KS QS 9S 6S 4S QH JH 5H 4H 2H AD QC 5C
This hand is worth 17 points.
Running simulation.....

The estimated probability based on 500 simulated hands:
Pass: 1.2%
Part score: 30.8%
Game: 52.8%
Small Slam: 12.4%
Grand Slam: 2.8%

Another hand [Y/N]? y
Here is your hand:
KS 8S 6S 5S KH 10H 7H 5H 3H 7D JC 8C 6C
This hand is worth 9 points.
Running simulation.....

The estimated probability based on 500 simulated hands:
Pass: 30%
Part score: 49.6%
Game: 19%
Small Slam: 1.4%
Grand Slam: 0%

Another hand [Y/N]? n

C:\Users\Brian\Documents\Visual Studio 2019\Projects\Bridge
 exited with code 0.
```

3 The RNG was originally a *linear congruential operator*, a function of the form $y_{n+1} = ((y_n*k)+c) \bmod M$. For carefully selected values of k, c, and M, it is possible to produce a sequence of maximal length; that is, each value from 0 to M-1 appears exactly once before any value repeats. In effect, it defines a permutation of the integers from 0 to M-1, and a particular seed value just specifies the starting point within that permutation. For Visual Studio, the value produced was then reduced via modulus again to a 15-bit unsigned integer, to maintain backward compatibility.

Appendix: The Standard Deck for Bridge

The standard deck consists of 13 cards in each of 4 suits. The suits are: Spades, Hearts, Diamonds, Clubs. The ranks are: Ace (highest), King, Queen, Jack, 10, 9, 8, 7, 6, 5, 4, 3, 2 (lowest).

In Bridge, the suits are ranked, in the order listed above; Spades and Hearts are the ‘Major’ suits, Diamonds and Clubs the ‘Minor’ suits. We do not need to worry about that here. Likewise, we do not need to worry about the intricacies of Bridge bidding, only a rough evaluation of the quality of a hand.

There are no ‘wild’ cards, and the Ace is always high, the 2 always low. Each player is dealt 13 cards. The order the cards are dealt in does not matter (it’s a combination, not a permutation).